AI Workshop – Lecture 1

A Brief Introduction to Caffe & Linux

1.0 Introduction

There are a number of modern frameworks for neural networks. There are two purposes for these frameworks:

- To aid the development of neural networks by humans (some popular examples are Tensorflow, Keras, Theano and Caffe)

- To efficiently run (ie train) these networks on GPUs (Graphical Processing Units) and CPUs. (examples are the opensource BLAS & Intel's MKL libraries and CuDNN) Of these, CuDNN specialises in providing highly tuned primitives for Nvidia's GPU products, while Intel's MKL libraries offer very fast performance on the CPU.

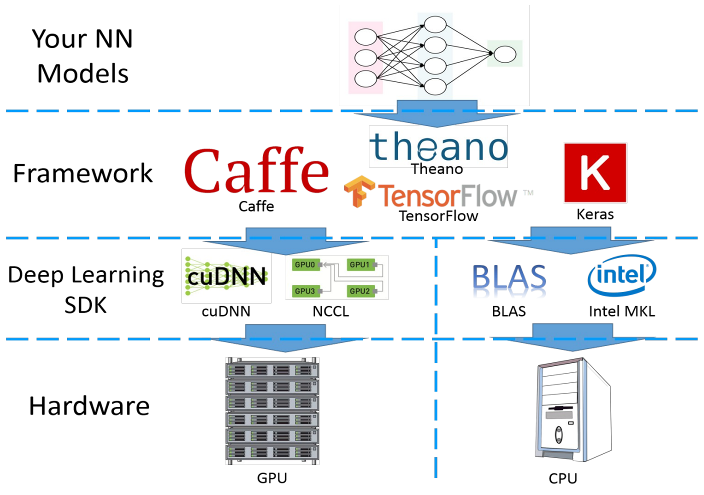


Figure 1.0 : Typical NN setup.

You won't need to install any of these software, since we have prepared the software infrastructure needed to run NNs on Caffe. You will need to use VirtualBox and our linux disk. This will be done in today's first Lab session.

In this course, we will work on CPU-only development, since not all laptops come installed with a Nvidia CUDA-enabled GPU. So, the stack we use is Caffe / Atlas (BLAS) / CPU.

1.1 Caffe

Caffe is an early NN framework, developed for Machine Vision applications. We advocate using Caffe because:

1. **It is fast**: Certainly it is among the fastest NN frameworks we have tested to date. Speed is necessary since we will need to run many experiments numerous times.

2. **It offers a declarative interface for NNs**: Rather than writing Python code (required for Tensorflow / Keras / Theano), Caffe offers a much simpler, faster (but less flexible) format called **prototext**. So, you can design your NNs without needing to program Python. We will cover the prototext format gradually in the Lab sessions over the next few days.

3. **It is easy to "hack"**: It is relatively easy to add new C++ and CUDA-C layers into Caffe. Other networks offer Python, but this is still much slower than C++, at least for now. The Terra Weather team have written a number of new open source additions to Caffe (see http://ai.terrawx.com) for timeseries predictions which will be introduced and used in this course. These layers are **not** blackboxes, but will be explained in detailed in this course.

4. **It is easy to automate**: Automation makes running numerous experiments easy. We have written an automation system for Caffe (which we've called Autocaffe) which will be used in this course to simplify and speed up NN development to solve real-world problems.

1.2 Autocaffe

Autocaffe is a simple tool we use internally in Terra Weather to:

1. Automatically schedule NNs for running,
2. Automate the collection of performance statistics,
3. Automate visualization of numerous experiments in a single PDF report,
4. Automatically create and run NNs having different configuration parameters,
5. Automatically generate NNs from pre-defined templates (called prefabs). This can greatly simplify creating more complicated networks like stacked autoencoders.

Autocaffe is a productivity tool. It is useful only if you need to systematically try many different configurations or if you need to train complex networks, as you will likely need to for this Datathon. To the best of our knowledge, there are no comparable tools like this for Tensorflow/Keras/Theano.

1.3 Other tools

For this course, we encourage you to use a popular Python-based data exploration tool called Jupyter Notebook. This is a great tool for doing initial data exploration since it allows you to run Python code and plot graphs quite easily. This is very helpful in deciding which input pre-processing is useful, prior to implementing it in Caffe. This aspect will be covered in future Lectures.

1.3.1 Linux

One of the easiest development environments for NN projects is Linux. The main advantages of using Linux for NN development are:

- Excellent compilers (both free and paid) for Linux. These are needed for compiling changes to your NN framework.
- Many highly reliable open-source packages for Linux. Many of these are used in Caffe.
- Oftentimes, much easier installation of various libraries needed for NN work. This is true for Caffe,Tensorflow and Keras.
- A powerful command line shell called Bash. This greatly simplifies automating Caffe.
- The built-in ability to perform scheduling and many other tasks.
- Linux is free. You can with little effort build a high-performance system for free.

Other popular OSes offer these to some degree, but often change the behaviour of these tools too rapidly for them to be relied upon. Where Linux shines is in providing a widely-used, well-tested, well-documented and very powerful NN development environment.

Typically, you would run your NN models on a powerful “headless” (ie, no-GUI) Linux server, on which is attached one or more GPUs. You would then run a standard VPN tool called SSH (**S**ecure **SH**ell) to submit jobs from your PC/lsptop to the server.

In this course, we will instead rely on your laptop to run your NNs. In order to do this, we will run a Linux Virtual Machine (VM) on your laptop. This VM will run a fully-fledged version of Ubuntu 17 (a very popular distribution of Linux), hosted within Window or MacOS.
Running NNs this way is far from ideal, since it will incur a 20% - 30% performance penalty, but it is more than offset by:

- Helping you get comfortable using Linux for NN development. We feel this is crucial for NN work.

- Allows us to run Terra Weather's Autocaffe , a powerful new automation for Caffe, which will be a great help to you in completing the Datathon.
- Help you work together with your team-mates, since you have a common development platform.
- Many popular computing platforms (eg Amazon's AWS, Google Cloud, Microsoft's Azure, etc) are actually virtualized Linux environments.

Lastly, while running VirtualBox will give you a significant performance penalty, we found running NNs on popular free cloud platforms give a 10,000% performance penalty, which makes these free offerings unhelpful for NN work in this course and the Datathon.

Lab 1

The purpose of this Lab session is to set up and familiarize you with the Linux environment. We will:

1. first ensure VirtualBox is running on your laptop,

2. then import our pre-built Linux disk. This comes with Python, TerraWeather's update of Caffe and Autocaffe, plus many other necessary software.

You will run VirtualBox as a “server” on your laptop (which we will call the “host”). As the server behaves like an independent machine, with its own operating system and programs, we will need to be able to interact with it. You will learn how to:

- transfer files to/from the “server” and your laptop's OS (known as the “host”),
- write and execute a simple program on the server.
- Finally we introduce the Jupyter Notebook, a very useful python-based tool for data processing and visualization.

Lab 1.1 – Setting up VirtualBox

Step 1: Using the link:

https://www.virtualbox.org/wiki/Downloads

Download and install Oracle VM VirtualBox version 5.2. If you have already set up VirtualBox using the guide sent out earlier, please skip this Step 1.

Step 2: Download our virtual machine “appliance” using this link:

https://1drv.ms/u/s!An58mTz89jfxjGZM7Y2h1DlNxKoo

This is a large file (2.3GB) and should be named Ubuntu.ova. This file is known as an “appliance”.

Step 3: Import appliance in VirtualBox, by doing the following:

1. File > Import Appliance
2. Browse for downloaded virtual machine and import. (see Fig L1.1.1)

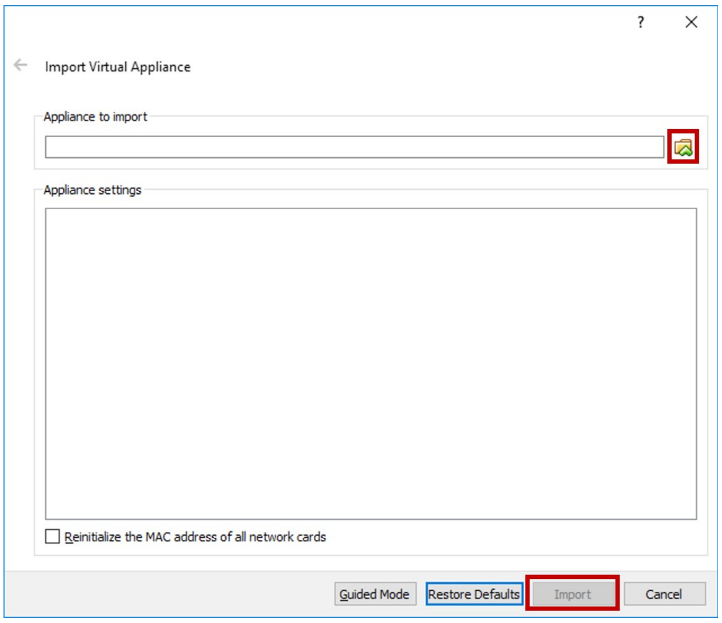
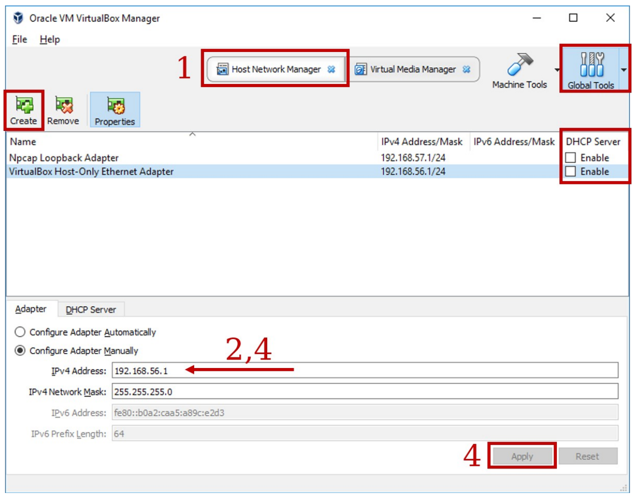


Figure L1.1.1 – The Import Appliance dialog

Step 4: We next need to set up the networking for our Appliance.

1. Select the 'Global Tools' button, then select 'Host Network Manager'
2. There should be an entry in the network list named 'Npcap Loopback Adapter'. Ensure that its IPv4 Address is <u>not</u> 192.168.56.1 by changing it to something else (eg. 192.168.57.1)
3. If there is only 'Npcap Loopback Adapter' in the list, click on 'Create' to create a new host-only network.
4. Select the second network in the list ('VirtualBox Host-Only Ethernet Adapter' in Figure L1.1.2 image below) and set its IPv4 address to 192.168.56.1. <u>Remember to apply the changes.</u>
5. Click on the checkboxes to <u>**disable**</u> the DHCP Server for both networks



Name	IPv4 Address/Mask	IPv6 Address/Mask	DHCP Server
Npcap Loopback Adapter	192.168.57.1/24		☐ Enable
VirtualBox Host-Only Ethernet Adapter	192.168.56.1/24		☐ Enable

Configure Adapter Automatically [✓ Configure Adapter Manually]
IPv4 Address: 192.168.56.1
IPv4 Network Mask: 255.255.255.0
IPv6 Address: fe80::b0a2:caa5:a89c:e2d3
IPv6 Prefix Length: 64

Figure L1.1.2 – The Host Network Manager dialog

Step 5: Virtual Machine settings

1. Click on ‘Machine Tools’ button
2. Select the Ubuntu virtual machine and click on ‘Settings’ (see Figure L1.1.3).

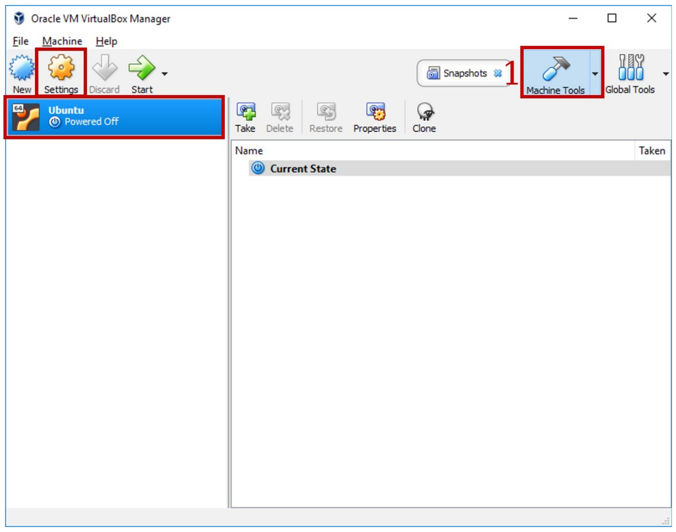
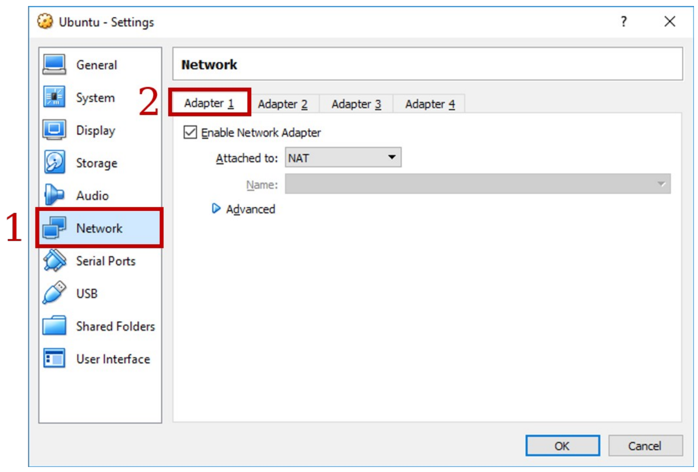


Figure L1.1.3 – The Machine Tools dialog

Step 6: Set up Network Adapter 1

1. In the settings window, select ‘Network’ in the list on the left

2. Select the 'Adapter 1' tab and ensure that the settings are the same as in Figure L1.1.4 below:



Network

☑ Enable Network Adapter

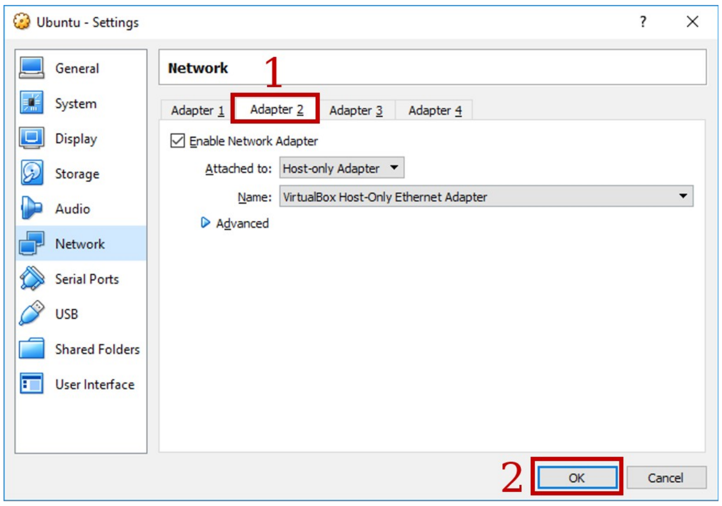
Attached to: NAT

Name:

Figure L1.1.4 – The Network Adapter 1 Dialog

Step 7: Set up Network Adapter 2

1. Select the 'Adapter 2' tab and change the settings to match Figure L1.1.5 below. The network name selected should be the second network listed in the Host Network Manager as set up earlier.
2. Click 'OK' to exit the settings



Network

☑ Enable Network Adapter

Attached to: Host-only Adapter

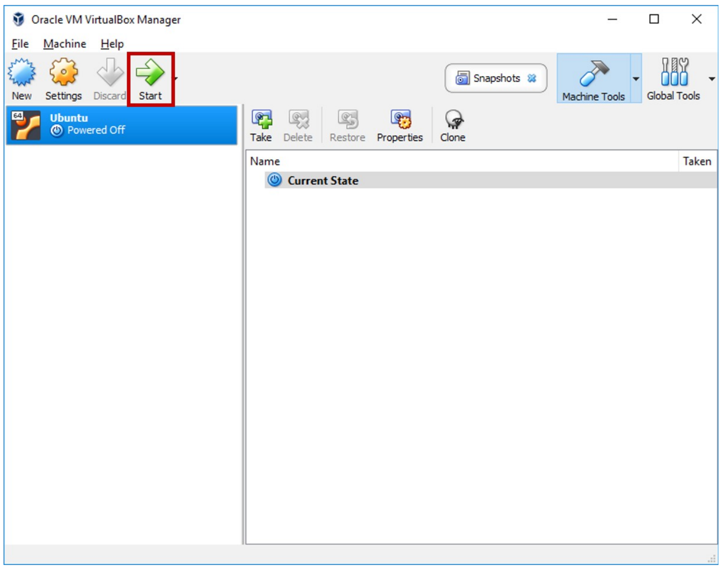
Name: VirtualBox Host-Only Ethernet Adapter

Advanced

Figure L1.1.5 - The Network Adapter 2 Dialog

Step 8: Start the Virtual Machine

Click the ‘start’ button (Figure L1.1.6 below):



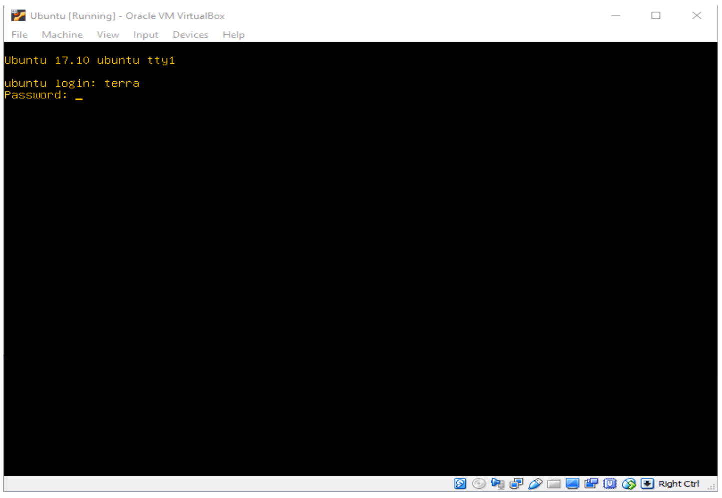
Ubuntu
Powered Off

Current State

Figure L1.1.6 – Starting the Virtual Machine server

Wait for Ubuntu to start up. You should see the black "headless" (ie, no-GUI) login screen (Figure L1.1.7 below) and login using the username: **terra** and password: **terran**

Note that the password will not appear as you type, just press enter when you're done. Enter the command **`screenfetch`** to see some details of the machine that you've just set up


Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Ubuntu 17.10 ubuntu tty1

ubuntu login: terra
Password: _
```

Figure L1.1.7 – Ubuntu headless login screen.

To shutdown the server, go to:

File > Close > Send the shutdown signal > OK

Alternatively, you can type in the command into the Ubuntu terminal:

sudo /sbin/shutdown now -h

This will prompt you again for the password (terran).

You <u>must</u> shutdown everytime you wish to stop a running server. Failure to do so will corrupt your Virtual Machine appliance.

Lab 1.2 – File transfers
To transfer files to and from your server, we use either the SCP (Secure Copy Protocol) or SFTP (SSH File Transfer Protocol) protocols. This can be done through the command line using the **scp** and **sftp** commands (see Appendix 1.A), however we recommend using a SFTP client as this is more convenient.

Instructions

<u>For Microsoft Windows:</u>
Windows users can choose between many SFTP clients. We will go with WinSCP which provides an easy way of editing files on the server.

Step 1: Download and install WinSCP:

https://winscp.net/download/WinSCP-5.11.2-Setup.exe

Step 2: At the login prompt, enter the settings as in Figure L1.2.1 below. Click on 'save' for future convenience, then click on 'login'

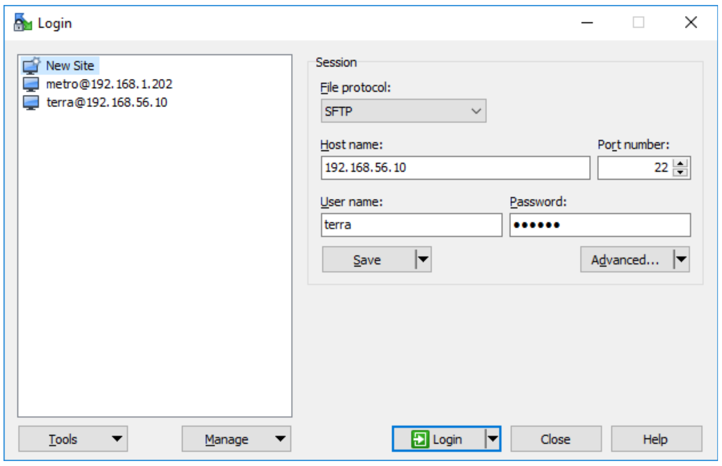
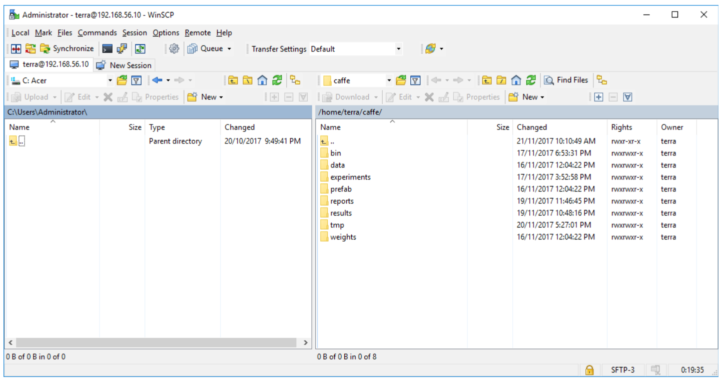

Session
File protocol: SFTP
Host name: 192.168.56.10
Port number: 22
User name: terra
Password: ••••••

Figure L1.2.1 – The WinSCP login Screen

The WinSCP interface is divided into 2 panes (refer to the Figure L1.2.2 below). On the left is a directory of the host, while the right side shows the server. You can navigate the file directory hierarchy by double clicking or hitting enter, just like in windows explorer. To transfer files, simply drag the file from one side to the other.



Administrator - terra@192.168.56.10 - WinSCP

terra@192.168.56.10

C:\Users\Administrator\

Name	Size	Type	Changed
..		Parent directory	20/10/2017 9:49:41 PM

/home/terra/caffe/

Name	Size	Changed	Rights	Owner
..		21/11/2017 10:10:49 AM	rwxr-xr-x	terra
bin		17/11/2017 6:53:31 PM	rwxrwxr-x	terra
data		16/11/2017 12:04:22 PM	rwxrwxr-x	terra
experiments		17/11/2017 3:52:58 PM	rwxrwxr-x	terra
prefab		16/11/2017 12:04:22 PM	rwxrwxr-x	terra
reports		19/11/2017 11:46:45 PM	rwxrwxr-x	terra
results		19/11/2017 10:48:16 PM	rwxrwxr-x	terra
tmp		20/11/2017 5:27:01 PM	rwxrwxr-x	terra
weights		16/11/2017 12:04:22 PM	rwxrwxr-x	terra

0 B of 0 B in 0 of 0

0 B of 0 B in 0 of 8

SFTP-3 0:19:35

Figure L1.2.2 – The WinSCP Main Window.

<u>For MacOSX:</u>
MacOSX already has built into it command-line SCP and SFTP software. You need to use these from the Terminal (see Figure L1.2.3 below)

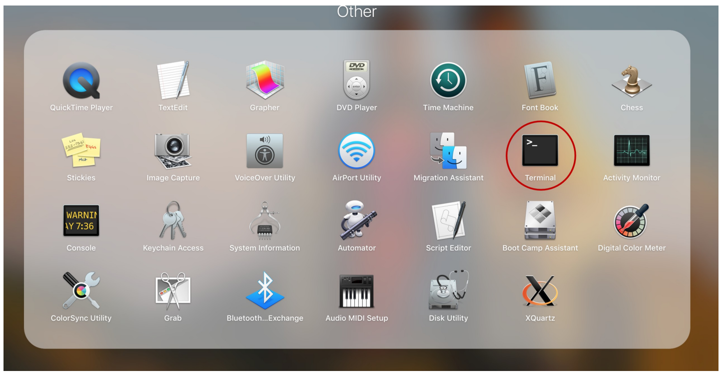


Figure L1.2.3. Mac Terminal

You can type in commands directly into the Terminal. This is the recommended way to access the Virtual Machine server on OSX.

However, if you wish to use a GUI, note that WinSCP is not available on OSX, we provide instructions for FileZilla. You may use any other SFTP client of your choice.

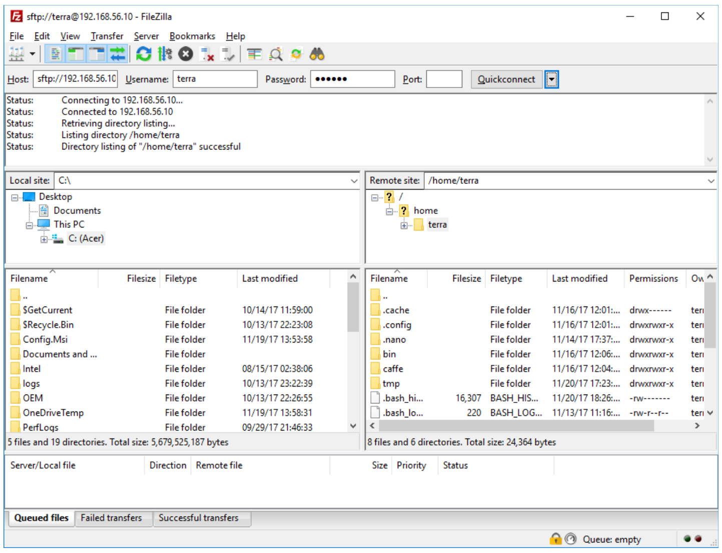
1. Download and install FileZilla:

   https://filezilla-project.org/download.php?type=client

   NOTE: FileZilla is bundled with other unrelated software. Be careful to uncheck the options to install these software during the installation process!

2. Run FileZilla, fill in the login information at the top of the window, then select “Quickconnect”

The FileZilla interface is split into two panels. On the left is the directory structure for the host machine, while on the right is same for the server. Navigate the directory hierarchy by double clicking on folders or using the directory tree. To transfer files, simply drag the relevant files from one side to the other.


sftp://terra@192.168.56.10 - FileZilla
File Edit View Transfer Server Bookmarks Help
Queued files
Failed transfers
Successful transfers
Queue: empty

Host: sftp://192.168.56.10 Username: terra Password: •••••• Port: Quickconnect

```
Status:     Connecting to 192.168.56.10...
Status:     Connected to 192.168.56.10
Status:     Retrieving directory listing...
Status:     Listing directory /home/terra
Status:     Directory listing of "/home/terra" successful
```

Local site: C:\

```
Desktop
  Documents
  This PC
    C: (Acer)
```

Remote site: /home/terra

```
/
  home
    terra
```

Filename	Filesize	Filetype	Last modified
..			
$GetCurrent		File folder	10/14/17 11:59:00
$Recycle.Bin		File folder	10/13/17 22:23:08
Config.Msi		File folder	11/19/17 13:53:58
Documents and ...		File folder	
Intel		File folder	08/15/17 02:38:06
logs		File folder	10/13/17 23:22:39
OEM		File folder	10/13/17 22:26:55
OneDriveTemp		File folder	11/19/17 13:58:31
PerfLogs		File folder	09/29/17 21:46:33

5 files and 19 directories. Total size: 5,679,525,187 bytes

Filename	Filesize	Filetype	Last modified	Permissions	Ow
..					
.cache		File folder	11/16/17 12:01:...	drwx------	ter
.config		File folder	11/16/17 12:01:...	drwxrwxr-x	ter
.nano		File folder	11/14/17 17:37:...	drwxrwxr-x	ter
bin		File folder	11/16/17 12:06:...	drwxrwxr-x	ter
caffe		File folder	11/16/17 12:04:...	drwxrwxr-x	ter
tmp		File folder	11/20/17 17:23:...	drwxrwxr-x	ter
.bash_hi...	16,307	BASH_HIS...	11/20/17 18:26:...	-rw-------	ter
.bash_lo...	220	BASH_LOG...	11/13/17 11:16:...	-rw-r--r--	ter

8 files and 6 directories. Total size: 24,364 bytes

Server/Local file	Direction	Remote file	Size	Priority	Status

Figure L1.2.4 – The Filezilla Main Screen

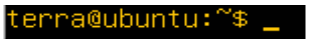
Lab 1.3 – Bash scripting

Bash is the name of the program that interprets the commands that you type into the Linux command line. Such commands include `cd` for changing the directory, `cp` to copy a file (see Appendix 1.A for more commands and how to use them). Groups of commands that are frequently used can be pre-written and saved together as a script, which by convention has the `.sh` extension. These scripts can then by run directly or be called by other scripts. We will demonstrate the basic capabilities of Bash by writing a simple script.

Instructions

Start the Ubuntu virtual machine in VirtualBox and log in.

Option 1: (Windows) After logging in, you will be shown this line:



```
terra@ubuntu:~$ _
```

This is the **command line**, and whatever you type in here will be read and interpreted by Bash. The line starts with the username (`terra`) and server name (`ubuntu`), shown separated by the `@` character. The current directory is then given before the dollar sign `$`. In this case, we are in the directory ~. In Bash, ~ is short for the home directory (`/home/terra`). The current directory can also be found by using the command `pwd`.

Option 2: (MacOSX/Linux) Open up a Terminal, then use SSH to log into the server from the Terminal:

```
ssh terra@192.168.56.10
```

You will be prompted for the password (terran) and you should then get a welcome screen. Upon successful login, any further commands will be executed on the server, not on the Host. Be always aware of where you are typing in commands (especially with “dangerous” commands like **rm**).

Changing directory

To navigate to another directory, we use the `cd` command. `cd` can accept both relative or absolute paths. Absolute paths refer to the full path name of the directory you want to go to (eg. `cd /home/terra/bin`) while relative paths are relative to the current directory (eg. `cd bin`). Since we are currently in the home directory, executing `cd bin` will lead to `/home/terra/bin`. To go back to the home directory, do a `cd ~` (absolute path) or `cd ..` (relative path), where the double dots `..` means the parent directory. Remember that you can always check your current directory using `pwd`.

Tip: You can reduce typing by making use of the autocomplete function of Bash. Just hit `tab` after typing the first letter of your target folder that uniquely determines it. For example you can type `cd /h <tab> <tab> b <tab>` to navigate to `/home/terra/bin`. The second `<tab>` is able to expand to `terra/` because that is the only file or directory that exists in `/home`.

Making a script

We now create a simple script that reads user input and displays some text. Start by going to the home folder and entering `vim hello.sh` . This command opens the VIM text editor and tells it to make a file called `hello.sh`. The VIM editor starts in its command mode, in which keypresses are treated as VIM commands. To start editing the file, you have to switch to the insert mode by hitting the key `i`. You can then begin typing.

Copy the following lines into your script and follow the explanations of each line.

```
#!/bin/bash
```

This line is known as the “shebang”. It provides the path to the interpreter, Bash in this case. This must be the first line of all Bash scripts.

```
echo “What is your name?”
```

`echo` is a command that displays its arguments on screen, in this case the request for your name.

```
read name
```

The `read` command waits for user input and stores that input into the variable `name`.

```
echo “Hi $name! Can we be friends?”
```

This line shows how you can use variables. Bash substitutes `$name` with the contents of the name variable. The dollar sign indicates to Bash that `name` is a variable.

To save the file, we need to return to the command mode in VIM. Do this by pressing `esc`, then save by typing `:wq` . If you want to quit without saving, use the command `:q!`

Script execution

Now try to run the script by typing `./hello.sh` . The single dot means the present directory – so `./hello.sh` refers to the file `hello.sh` in the current directory.

You should encounter a permission error. In Linux, each file has 3 types of permissions – read, write and execute – that determines whether one can perform an operation on the file. Newly created files do not automatically have the right to be executed, hence the permission error. To change the permission we use the `chmod` (change file mode bits) command by running

```
chmod +x hello.sh
```

In the above command, the `+x` tells `chmod` that you want to add (`+`) the execute permission (`x`) to the file named `hello.sh`. Similarly, you can also remove (`-`) the read (`r`) or write (`w`) permission.

Try running your script again. Now you have a tool for extending a personalized friendship invitation! There is much more to Bash scripting that cannot be covered in this short exercise. Bash scripts allows you to easily write programs to automate repetitive tasks or manipulate files and is an integral part of working in Linux.

Lab 1.4 – Jupyter notebook

Jupyter notebook is a widely-used application that allows you to mix explanatory text and equations, python code and their output, all in a single notebook format. This is extremely useful for performing data analysis and interactively presenting your results, whether they are graphical plots, tables or individual numerical values.

Instructions

Create a Folder

Let’s start by making a folder to store all the notebooks that will be created. First go to the home directory (`cd ~`) and use the `mkdir` (make directory) command

```
mkdir notebooks
```

You can check that the directory has been made by using the `ls` command that lists the contents of your current directory. Then `cd` into the `notebooks` folder.

Start Jupyter notebook
To start the notebook server on your virtual machine, simply run

```
jupyter notebook
```

Then open up a browser (ie. Chrome, Firefox) **on your host system (not on the VM)** and type in the address bar

```
192.168.56.10:8888
```

This launches the Jupyter Notebook App, which is served by the notebook server running on the virtual machine. The Notebook App starts with a dashboard that allows you to navigate the folder structure and create/rename/delete files. Since the Jupyter notebook *server* was run from the newly-created notebooks folder, the dashboard displays an empty list of files.

Create a python notebook by clicking on the "New" button on the top right of the web interface, and select the "Python 3" option from the drop menu.

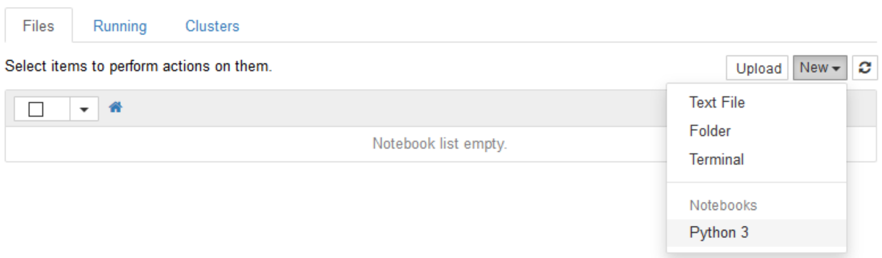
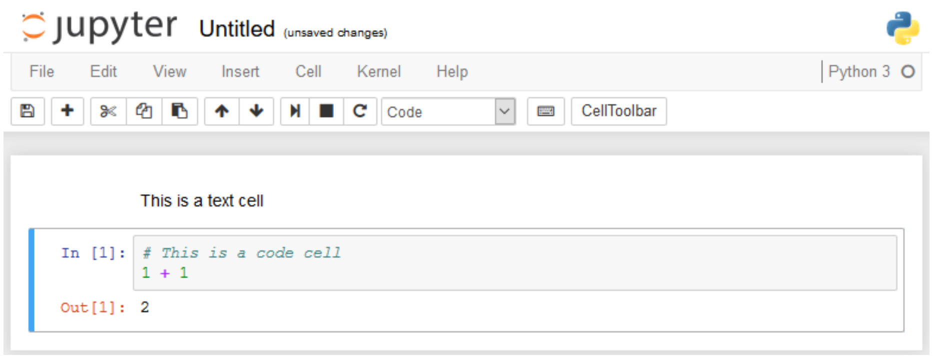


Figure L1.4.1 – Creating a new python notebook

Cell types
As mentioned in the introduction, Jupyter notebooks can contain rich textual information, as well as code blocks. Textual data are stored in 'Markdown' cells while code is stored in 'code' cells



This is a text cell

```
In [1]:  # This is a code cell
         1 + 1
Out[1]:  2
```

Figure L1.4.2 – Jupyter notebook with some cells

Markdown cells are so called as they use the Markdown markup language, which has a simple syntax to learn. For example to italicize text, just surround it with *, ie

```
normal text *italicized* normal text
```

For bold text, use double stars, `**` instead of single as for italics. Latex equations can also be added into markdown cells by wrapping the latex code with `$` for inline equations, and `$$` for equations on their own line.

You can refer to a quick guide to the Markdown syntax here.

Code cells contain python code. (sidenote: Jupyter notebook actually supports over 40 different languages, but we will only use python notebooks here) When a cell is run (by pressing shift+enter or ctrl+enter) the code in the cell is executed and any output is displayed right below the code textbox, but is still part of the same cell.

Within a notebooks. the state, ie the values of variables, are shared among all the cells. So **executing cells in different order may change their outputs** if they rely on variables set in another cell. The state is also not stored in the notebook files. This means when you first open a notebook, it has an empty state until you manually start executing code cells.

Refer to the table below on how to perform common cell operations. The most useful functions have buttons on the toolbar – be sure to familiarize yourself with them.

Creating new cells	Insert > Insert Cell Above / Below, or Second button on the toolbar
Cut/Copy/Paste/Delete cell	All these operations have their own buttons on the toolbar and can also be accessed from the Edit menu
Change cell type	Choose the desired cell type from the drop menu in the toolbar
Select multiple cells	Shift+click on the cells *outside the textbox*
Execute cell	Ctrl+enter
Execute cell and select the next cell	Shift+enter
Save notebook	First button on the toolbar
Exit notebook	File > Close and Halt

Plotting a sine curve
As an example on how to use Jupyter notebooks, we'll make a plot of a sine curve.

We will use the first cell for the title, so change the cell to to be a Markdown cell and enter this line

```
# Sine Wave
```

before executing it (ctrl+enter or shift+enter). The `#` mean that the text is a header. You can try it with multiple `#` and see what happens.

Now create another cell to input the plotting code. (If you had used shift+enter to execute the first cell, then the second cell would already have been created)

Copy the following python code (or write your own!):

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
plt.plot(x, y);
```

To give a brief explanation of the code:

- Lines 1 & 2: imports the required math and plotting libraries
- Line 3: `%matplotlib inline` is required for the plot to be displayed in the notebook
- Line 4: store 100 equally-spaced numbers from 0 to 2π in the variable `x`.
- Line 5: calculate the sine of all the values in `x` and stores them in variable `y`
- Line 6: plot `y` against `x`.

Try doing other calculations, eg. plotting the number of values in `y` that are above a certain threshold value against the threshold value itself. You may want to take a look at this link and get inspired by the wide range of work that is possible with Jupyter notebooks.

After you are done, save and close the notebook. Then stop the notebook server by pressing `ctrl+c` at your virtual machine.

Appendix 1.A: Linux Command Cheat Sheet

This is a list of commonly used Linux commands. Note that all commands must be in lower case.

Command	Meaning & Usage
`man`	**Man**ual: Allows you to easily look up tha manual page for any command. Scroll throught the manual with the arrow keys, and quit by pressing `q`. Example: `man cp`
`pwd`	**P**rint **w**orking **d**irectory: Prints the current working directory on screen

`cd`	**C**hange **D**irectory: Allows you to navigate a file directory hierarchy. Examples:
	
	`cd ~`
	Changes to the user's home directory.
	
	`cd ..`
	Goes to the parent directory
	
	`cd ~/experiments/samuel`
	Goes to the sub-directory “samuel” within the “experiments” directory in the user's home.

`ls`	**Lis**t: Lists the files in the current directory

`cp`	**Co**p**y**: Used for copying of files or directories to another location. Examples:
	
	`cp file1.txt MyFiles/`
	Copies a file to the directory MyFiles
	
	`cp file1.txt MyFiles/newname.txt`
	Copes and renames a file at the new location

	`cp -r folder1 ~` Copies folder1 to the home directory

`mv`	**Mov**e: Moves files or folders from one location to another. Examples:
	
	`mv file1.txt ../MyFiles/`
	Moves file1.txt to the MyFiles directory in the parent directory
	
	`mv file1.txt file2.txt`
	Renames file1.txt to file2.txt

`rm`	**Rem**ove: Deletes files or folders. Examples:
	
	`rm *.txt`
	Deletes all files in the current directory with names ending with ".txt"
	
	`rm -r MyFiles`
	Deletes the directory named MyFiles

`mkdir`	**M**a**k**e **dir**ectory: Creates a new directory. Example: `mkdir FolderName`
`chmod`	**Ch**ange file **mod**e bits: Allows you to set the permitted operations on a file. Examples `chmod +x *.py` Allows execution of all python scripts in the current directory
`ssh`	SSH client: Allows you to login to a remote machine and execute commands. Example: `ssh terra@192.168.56.10`

	Login as user terra at 192.168.56.10

`scp`	**S**ecure **c**o**p**y: Copies files and directories between computers. Examples:
	
	`scp ~/file1 terra@192.168.56.10:~`
	Copies file1 from your home directory to the home directory of the user terra of computer 192.168.56.10
	
	`scp -r terra@192.168.56.10:~/myfolder .`
	Copies the directory myfolder from the home directory of the user terra of computer 192.168.56.10 to the current directory.

`ifconfig`	Configure network: Displays all operational network interfaces
`vim`	VIM text editor: Used for editing plain text files. Use the arrow keys to navigate the cursor. To do an edit, you have to switch to insert mode by pressing ‘`i`’. After the edits are done, switch back to the command mode by pressing the `Esc` key. Then you can save the file by typing `:wq` or `:q!` to quit without saving. VIM is full of features that allows you to work very efficiently with text. Refer to the many VIM tutorials online to learn more.